L8: Entry 2 of 5 File: USPT May 15, 2001

DOCUMENT-IDENTIFIER: US 6233688 B1
TITLE: Remote access firewall traversal URL

Abstract Text (1):
The invention provides a generic naming scheme for remote access and firewall traversal in the form of a uniform resource locator (RAFT URL). The RAFT URL may be provided to any client, regardless of compatibility with the remote access/firewall traversal method, which then launches an operating environment code module. The operating environment code module performs the remote access/firewall traversal method and interacts with the operating environment to obtain data transport mechanisms. These mechanisms permit the client application to transact with private resources beyond the firewall. The remote access/firewall traversal procedure is made transparent to the client application, and thus, a wider array of client applications may be chosen for the data session with the resources beyond the firewall.

Brief Summary Text (10):
The invention provides a generic naming scheme for remote access and firewall traversal in the form of a uniform resource locator (RAFT URL). The RAFT URL may be provided to any client application, regardless of compatibility with the remote access/firewall traversal method, which then launches another executable module. The executable module performs the remote access/firewall traversal method and interacts with the operating environment to obtain data transport mechanisms. These mechanisms permit the client application to transact with private resources beyond the firewall. The remote access/firewall traversal procedure is made transparent to the client application, and thus, a wider array of client applications may be chosen for the data session with the resources beyond the firewall.

Detailed Description Text (5):
Once on the Internet 130, remote access client 110 is free to access services provided by any computers or networks connected to the Internet 130 such as an HTTP service (i.e., a web site). However, the below description restricts itself to using the Internet to gain remote access into a private network or intranet 150 by means of the Internet. Once remote access into the intranet 150 is established, remote access client 110, services provided for within the intranet 150 may be accessed. What distinguishes the intranet 150 from an ordinary web server or FTP server on a more public network is the security and isolation that can be provided by a gateway or firewall 140. One function of firewall 140 is to prevent unauthorized access into the intranet by users/computers connected to the Internet 130.

Detailed Description Text (6):
To control and configure access to the intranet 150 through firewall 140 many various firewall protections or security schemes have been developed such as IP security schemes using SKIP (Simple Key Management for Internet Protocols), or ISAKMP (Internet Security Association and Key Management Protocol), SSL (Secure Sockets layer), etc. Many of these schemes are conflicting and standardization has not been successfully achieved. One reason for the failure of standardization is the nature of remote access--it is intended for a specific often closed set of users. For instance, a company X may desire that only certain key employees have remote access to the intranet of X. In that case, company X will choose whatever method of remote access security is easy to implement or whatever method is decided on as best for the type of information served. Since the choice of remote access security

WEST

Generate Collection Print

Norman

L2: Entry 2 of 5 File: USPT Mar 21, 2000

DOCUMENT-IDENTIFIER: US 6041041 A
TITLE: Method and system for managing data service systems

Brief Summary Text (14):
Another drawback is that since modules are measured in isolation, their measurements do not assess the availability and performance of the modules as the availability and performance relate to the service being provided using the modules. For example, the performance measurements for a firewall measured in isolation are CPU utilization at the firewall, packet handling rate (i.e., packets per second) of the firewall, the delay introduced by the firewall in routing a packet, and the number of packets discarded by the firewall per second because of buffer overflows. However, the impact of the firewall performance on the ISS system depends on several factors that are external to the firewall. These factors include the specific TCP (Transmission Control Protocol), the TCP/IP stack used in users' terminal and in the web content and proxy servers, the TCP window size used by the web browser and web server application modules, and the size of the data transfer. Furthermore, the location of the firewall in the topology of the ISS system determines which of the functional modules the firewall impacts. Therefore, the precise impact of the firewall performance on the performance of the ISS system cannot be measured by considering the firewall in isolation.

Detailed Description Text (40):
Another advantage of the test target is that for some service topologies, the service topology may not permit sufficiently diverse measurement routes to be developed to diagnose or identify problem modules. In such cases, by placing test targets at strategic places in the system 60, problems may be isolated effective and efficiently. FIG. 17 shows an example. As can be seen from FIG. 17, tests can be run to the remote web server 500 of the other ISSs 56 (FIG. 3) via (1) the proxy server 67 and the firewall 63 and (2) the firewall 63 based on the service topology shown in FIG. 17. When measurements from both measurement routes show problems, there is no direct way to ascertain whether the problem is because of the remote web server 500 or because of the firewall 63. This means that the available measurement routes are not sufficient to either eliminate the firewall 63 as the problem module or positively identify it as the problem source. With the test target 72 associated with the firewall 63, an additional measurement route can be established to measure the impact of the firewall 63.

Detailed Description Text (45):
To assess the status of the system 60, the measurement system 70 uses the surveillance measurement routes alone. To minimize the overheads of the measurements on the system 60, the measurement system 70 selects a minimum number of measurement routes that cover all modules of the system 60. Those measurement routes, however, provide essential measurements for determining the overall status of the system 60. For example, as can be seen from FIG. 7, the measurement route (i.e., the route 113) that retrieves a web page from a local web server that is located in the other ISSs 56 provides information about the health of the proxy servers 67, the DNS servers 65, the firewall 63, and the local web server in the other ISSs 56. If the measured response time for retrieving web pages from the local web server in the other ISSs 56 is less than a pre-specified threshold, the DNS and proxy servers 65 and 67, the local web server, and the firewall 63 are functioning as expected. If the measured response time is greater than the pre-specified threshold, then all of the modules are suspect modules and more measurements are need to show the status and

performance of the individual modules, and to isolate the problem module or modules.

Detailed Description Text (47):
As can be seen from FIG. 8, these three measurement routes all pass through the firewall 63. The measurement route 90 measures the status of the other ISS's web server 56 via the firewall 63 and the proxy servers 67. The route 91 is only via firewall 63. The measurement route 92 measures the response time to an Internet web server Internet 55 via the firewall 63. Therefore, if the measurement of any one measurement route is good, the firewall 63 can be identified as a non-problematic module. If, on the other hand, one measurement route (e.g., the measurement route 90) indicates that the performance of the system 60 is not meeting the expectation, then measurements can be taken from the measurement routes 91 and 92 to find out if the firewall 63 is the bottleneck module. If the measurement from the measurement route 91 indicates good measurement results, then the firewall 63 can be identified as the non-problematic module. If the measurement route still indicates problem, then the measurement route can be taken to further determine if the firewall 63 can be isolated.

Detailed Description Text (49):
Another problem is indicated between 7:00PM and 9:00PM (see FIG. 9B). At this time, there are two common modules (i.e., the other ISS's web server 56 and the firewall 63) and more measurement routes are needed to isolate the problem.

Detailed Description Text (50):
To find out if the firewall 63 is performing as expected, the third measurement route 92 is used. As can be seen from FIG. 9C, the 130 130 indicates problematic performance or bottleneck at the same time. This basically identifies the firewall 63 as the most likely problematic module. As described above, the process of isolating the problem module is shown in FIG. 16, which will be described in more detail below.

Detailed Description Text (77):
FIG. 16 shows the diagnosis process for isolating or identifying problematic modules in the system 60 based on the status display of FIG. 15. FIG. 16 shows the algorithm for diagnosing the problem module using the dependency graph of FIG. 7. Diagnosis is simple if there is only one module in FIG. 15 that has a rating of 1 or more. In this case, the module with a rating of 1 or more is the problem module. When several modules have a rating of one or more, it is unclear whether all of those modules are problem modules or whether only a subset of those modules are problem modules. The complexity in diagnosis occurs because of the interdependencies between modules. As can be seen in the example of FIGS. 15, two modules, namely the proxy server 67 and the firewall 63 have ratings of one or more. From the service topology in FIG. 7, it is clear that a bottleneck firewall 63 may slow down all connections it handles, resulting in an accumulation of connections at the proxy server 67. Consequently, the proxy server 67 itself may appear to be slowing down in its performance. In this case, the degradation in performance of the proxy server 67 is attributable to the firewall 63. Hence, the firewall 63 is the real problem module, although the proxy server 67 also has a rating of 1 or more.

L8: Entry 3 of 5 File: USPT Apr 18, 2000

DOCUMENT-IDENTIFIER: US 6052788 A
TITLE: Firewall providing enhanced network security and user transparency

Abstract Text (1):
The present invention, generally speaking, provides a firewall that achieves maximum network security and maximum user convenience. The firewall employs "envoys" that exhibit the security robustness of prior-art proxies and the transparency and ease-of-use of prior-art packet filters, combining the best of both worlds. No traffic can pass through the firewall unless the firewall has established an envoy for that traffic. Both connection-oriented (e.g., TCP) and connectionless (e.g., UDP-based) services may be handled using envoys. Establishment of an envoy may be subjected to a myriad of tests to "qualify" the user, the requested communication, or both. Therefore, a high level of security may be achieved. The usual added burden of prior-art proxy systems is avoided in such a way as to achieve full transparency--the user can use standard applications and need not even know of the existence of the firewall. To achieve full transparency, the firewall is configured as two or more sets of virtual hosts. The firewall is, therefore, "multi-homed," each home being independently configurable. One set of hosts responds to addresses on a first network interface of the firewall. Another set of hosts responds to addresses on a second network interface of the firewall. In one aspect, programmable transparency is achieved by establishing DNS mappings between remote hosts to be accessed through one of the network interfaces and respective virtual hosts on that interface. In another aspect, automatic transparency may be achieved using code for dynamically mapping remote hosts to virtual hosts in accordance with a technique referred to herein as dynamic DNS, or DDNS.

Brief Summary Text (24):
The firewall may have more than two network interfaces, each with its own set of virtual hosts. Multiple firewalls may be used to isolate multiple network layers. The full transparency attribute of a single firewall system remains unchanged in a multi-layered system: a user may, if authorized, access a remote host multiple network layers removed, without knowing of the existence of any of the multiple firewalls in the system.

Detailed Description Text (22):
Referring more particularly to FIG. 3, there is shown a firewall 305 having a first set of virtual hosts 305a, a second set of virtual hosts 305b, and a DNS/DDNS module 315. The virtual hosts do not require and preferably do not have access to the disk files of the underlying machine. Instead, virtual host processes are spawned from a daemon process that reads a master configuration file from disk once at start-up. The DNS/DDNS module and the special-purpose virtual host 317 do have access to disk files 316 of the underlying physical machine. The special-purpose virtual host 317, shown in exploded view, runs an HTML-based Configurator module 319. Access to the special-purpose virtual host is scrutinized in accordance with rules stored on disk within configuration files 321. Typically, these rules will restrict access to a known secure host, will require at least username/password authentication and optionally more rigorous authentication. Once access is granted, the Configurator module will send to the authorized accessing host a first HTML page. From this page, the user may navigate through different HTML pages using a conventional Web browser and may submit information to the special-purpose virtual host. The special-purpose virtual host will then use this information to update the configuration files 321.

Detailed Description Text (28):
Channel processing may be performed using existing standard software modules. In the case of encryption and decryption, for example, modules for DES, RSA, Cylink, SET, SSL, and other types of encryption/decryption and authentication may be provided on the firewall. In the case of compression and decompression, standard modules may include MPEG, JPEG, LZ-based algorithms, etc. Based on information contained in the configuration files, information passing through the firewall may be processed using one or more such modules depending on the direction of data flow.

Detailed Description Text (35):
The configuration of FIG. 4, however, further allows the physical firewall machines 407 and 408 to share the aggregate processing load of current connections. Load sharing may be achieved in the following manner. Each of the DNS modules of all of the machines receive all DNS queries, because the machines are connected in parallel. Presumably, the DNS module of the machine that is least busy will be the first to respond to a query. An ensuing connection request is then mapped to a virtual host on the responding least-busy machine.

Detailed Description Text (36):
As the popularity and use of the Internet continues to grow, there is a concern that all available addresses will be used, thereby limiting further expansion. An important result of DDNS is that network addresses may be reused on network segments between which at least one firewall intervenes. More particularly, the addresses which are employed on opposite sides of a firewall are mutually exclusive of one another to avoid routing errors. Referring again to the example of FIG. 1, users of the Internet 120 are unaware of the addresses employed on a network segment 110. Certain addresses can be reserved for use behind a firewall. As shown in FIG. 1, for example, the subset of addresses represented as 192.168.X.X can be used on the network segment 110. So long as an address is not used on both sides of the same firewall, no routing errors will be introduced. Therefore, the same set of addresses can be used on the network segment 160, which is separated from the Internet via the firewall 157. On network segment 102 and network segment 152, the entire address space may be used, less those addresses used on the segments 110, 120 of the respective firewalls 105 and 155. Thus by isolating Internet Service Providers (ISPs) from the Internet at large using firewalls of the type described, each ISP could enjoy use of almost the full address space of the Internet (232 addresses). Exhaustion of network addresses, presently a grave concern within the Internet community, is therefore made highly unlikely.

End of Result Set

L8: Entry 5 of 5 File: USPT Apr 10, 1990

DOCUMENT-IDENTIFIER: US 4916704 A
TITLE: Interface of non-fault tolerant components to fault tolerant system

Abstract Text (1):
A fault tolerant computer system includes a fault tolerant data processing module which has means for detecting and correcting errors in the operation of the data processing module to maintain a high degree of data integrity. Data transmission control devices control the transmission of all data to the fault tolerant data processing module and the receipt of all data into the fault tolerant data processing module. Input/output terminals are coupled to the data transmission control means for receiving and transmitting data. A non-fault tolerant input/output module is coupled to transmit the data to the input/output terminals of the fault tolerant data processing module. This module includes a read device for transferring data to the fault tolerant computing system in response to requests from the data transmission control devices, and a firewall for preventing the non-fault tolerant input/output module from initiating transfers of data to the fault tolerant data processing module.

Brief Summary Text (2):
This application is related to the following U.S. patent applications all of which were filed on Sept. 4, 1987: U.S. Ser. No. 093,572 entitled "Dual Zone, Fault Tolerant Computer System With Error Checking on I/O Writes" by William F. Bruckert and Thomas D. Bissett; U.S. Ser. No. 093,584 entitled "Dual-Rail Processor With Error Checking at Single Rail Interfaces" by William F. Bruckert, Thomas D. Bissett and Norbert H. Riegelhaupt; U.S. Ser. No. 093,495 entitled "Dual-Rail Processors With Error Checking on I/O Reads" by William F. Bruckert and Thomas D. Bissett; U.S. Ser. No. 093,179 entitled "Fault Tolerant Computer System With Fault Isolation and Repair" by William F. Bruckert, Thomas D. Bissett, Dennis Mazur, John Munzer, Frank Bernaby and Jay H. Bhatia; U.S. Ser. No. 095,096 entitled "Synchronized Twin Computer System" by William F. Bruckert, Thomas D. Bissett, Dennis Mazur and John Munzer.

Brief Summary Text (10):
It is the object of the invention to provide a fault tolerant computer method and system having duplicate computer systems which normally operate simultaneously. The duplication insures that there is no single point of failure and an error or fault in one of the systems will not disable the overall computer system. Moreover, all such faults can be corrected by disabling or ignoring the module or element which caused the error.

Brief Summary Text (12):
To achieve these and other objects and in accordance with the purpose of the invention, as embodied and broadly described herein, there is provided a fault tolerant computer system comprising: a fault tolerant data processing module including means for detecting and correcting errors in the operation of said data processing module to maintain a high degree of data integrity; data transmission control means for controlling the transmission of all data to said fault tolerant data processing module and for controlling the receipt of all data into said fault tolerant data processing module, and input/output terminals, coupled to said data transmission control means, for receiving and transmitting said data; and a non-fault tolerant input/output module, coupled to transmit said data to said

input/output terminals of said fault tolerant data processing module, said non-fault tolerant input/output module including read means for transferring data to said fault tolerant computing system in response to requests from said data transmission control means; and firewall means for preventing said non-fault tolerant input/output module from initiating transfers of data to said fault tolerant data processing module.

Brief Summary Text (13):
In another aspect of the invention the fault tolerant data processing module includes a memory for storing data from the input/output module; and address means, coupled to the memory and under sole control of the fault tolerant data processing module, for generating addresses indicating the locations in the memory where the data is to be stored.

Brief Summary Text (14):
Yet another aspect of the invention provides that the input/output module includes an input/output bus for transferring information internal to the input/output module, and wherein the firewall means includes a bus interface element coupled to the input/output bus.

Brief Summary Text (15):
It is another aspect of the invention that the data transmission control means includes a cross-link element to control communication between the fault tolerant computer system and the non-fault tolerant input output module.

Brief Summary Text (16):
In another aspect, the invention provides a fault tolerant computer system comprising: a fault tolerant data processing module including means for detecting and correcting errors in the operation of said data processing module to maintain a high degree of data integrity; a memory unit for storing data for said fault tolerant data processing module; data transmission control means for controlling the receipt of all input data into said fault tolerant data processing system, said data transmission control means including means for generating all addresses in said memory unit into which data is to be received; input terminals, coupled to said data transmission control means, for receiving said input data; and an output module, coupled to transmit said input data to said input terminals of said fault tolerant data processing module, said output module including output means for transmitting said input data to said fault tolerant data processing module.

Drawing Description Text (5):
FIG. 3 is a block diagram of the CPU module shown in the fault tolerant computing system shown in FIG. 1;

Drawing Description Text (6):
FIG. 4 is a block diagram of an interconnected CPU module and I/O module for the computer system shown in FIG. 1;

Drawing Description Text (7):
FIG. 5 is a block diagram of a memory module for the fault tolerant computer system shown in FIG. 1;

Drawing Description Text (8):
FIG. 6 is a detailed diagram of the elements of the memory module shown in FIG. 5;

Drawing Description Text (9):
FIG. 7 is a block diagram of the primary memory controller of the CPU module shown in FIG. 3;

Drawing Description Text (10):
FIG. 8 is a block diagram of the mirror memory controller in the CPU module of FIG. 3;

Drawing Description Text (12):
FIG. 10 is a drawing of the parallel registers of the cross-link of the CPU module shown in FIG. 3;

Drawing Description Text (13):
FIG. 11 is a drawing showing the serial registers for the cross-link of the CPU module shown in FIG. 3;

Drawing Description Text (14):
FIG. 12 is a block diagram of the elements of the controller for the cross-link of the CPU module shown in FIG. 3;

Drawing Description Text (19):
FIG. 17 is a block diagram of an I/O module for the computer system of FIG. 1;

Drawing Description Text (20):
FIG. 18 is a block diagram of the firewall element in the I/O module shown in FIG. 17;

Drawing Description Text (29):
FIG. 27 is a logic flow diagram for isolation of intermittent faults for the computer system in FIG. 1; and

Drawing Description Text (30):
FIGS. 28A-28C is a logic flow diagram for isolation of solid faults in the computer system of FIG. 1.

Detailed Description Text (4):
FIG. 1 is a block diagram of a fault tolerant computer system 10 which achieves the objects of the present invention. Fault tolerant computing system 10 includes duplicate systems, called zones or stations. In the normal mode, the zones operate simultaneously. The duplication ensures that there is no single point of failure and that an error or fault in one of the zones will not disable computer system 10. Furthermore, all such faults can be corrected by disabling or ignoring the module or element which caused the fault. The two zones 11 and 11' are shown in FIG. 1 as including duplicate processing systems 20 and 20'. The duality, however, goes beyond the processing system.

Detailed Description Text (6):
As explained in greater detail below, processing systems 20 and 20' include several modules interconnected by backplanes. If a module contains a fault or error, that module may be removed and replaced without disabling computing system 10. This is because processing systems 20 and 20' are physically separate, have separate backplanes into which the modules are plugged, and can operate independently of each other. Thus modules can be removed from and plugged into the backplane of one processing system while the other processing system continues to operate.

Detailed Description Text (7):
The duplicate processing systems 20 and 20' are identical and contain identical modules. Thus, only processing system 20 will be described completely with the understanding that processing system 20' operates equivalently.

Detailed Description Text (8):
Processing system 20 includes CPU module 30 which is shown in greater detail in FIGS. 3 and 4. CPU module 30 is interconnected with CPU modules 30' in processing system 20' by a cross-link pathway 25 which is described in greater detail below. Cross-link pathway 25 provides data transmission paths between processing systems 20 and 20' and carries timing signals to ensure that processing system 20 and 20' operate synchronously.

Detailed Description Text (9):
Duplicative processing system 20 also includes I/O modules 100, 110, and 120, which are shown in greater detail in FIGS. 3 and 17. Each of the I/O modules 100, 110 and 120 is connected to CPU module 30 by dual rail module interconnects 130 and 132. Module interconnects 130 and 132 act as a backplane for processing system 20.

Detailed Description Text (13):
Generally, the two fail stop processing systems 20 and 20' operate in lock step

synchronism. There are three significant exceptions. The first is at initialization when a bootstrapping technique, explained in detail below, brings both processors into synchronism. The second exception is when the processing systems 20 and 20' operate independtly (asynchronously) on two different workloads. The third exception occurs when certain errors arise in processing systems 20 and 20'. In this last exception, one of processing systems or modules is disabled, thereby ending synchronous operation.

Detailed Description Text (14):
The synchronism of duplicate processing systems 20 and 20' is implemented by treating each system as a deterministic machine which, upon receipt of the same inputs and starting in the same known state, will always enter the same machine state and produce the same results unless there is some error. Processing systems 20 and 20' are configured identically, receive the same inputs, and therefore pass through the same states. Thus, as long as both processors operate synchronously, they should produce the same results and enter the same state. If the processing systems are not in the same state or produce different results, it is assumed that one of the processing systems 20 and 20' has faulted. The source of the fault must then be isolated in order to take corrective action, such as disabling the faulting module.

Detailed Description Text (15):
Error detection generally involves overhead in the form of additional processing time or logic. To minimize such overhead, a system should check for errors as infrequently as possible consistent with fault tolerant operation. At the very least, error checking must occur before data is outputted from CPU modules 30 and 30'. Otherwise, internal processing errors may cause improper operation in external systems, like a nuclear reactor, which is the condition that fault tolerant systems are designed to prevent.

Detailed Description Text (16):
There are reasons for additional error checking. For example, to isolate faults or errors it is desirable to check the data received by CPU modules 30 and 30' prior to storage or use. Otherwise, when erroneous stored data is later accessed and additional errors result, it becomes difficult or impossible to find the original source of errors, especially when the erroneous data has been stored for some time. The passage of time as well as subsequent processing of the erroneous data may destroy any trail back to the source of the error.

Detailed Description Text (19):
C. Module Description

Detailed Description Text (20):
1. CPU Module

Detailed Description Text (21):
The elements of CPU module 30 which appear in FIG. 1 are shown in greater detail in FIGS. 3 and 4. FIG. 3 is a block diagram of the CPU Module, and FIG. 4 shows block diagrams of CPU module 30 and I/O module 100 as well as their interconnections. Only CPU module 30 will be described since the operation of and the elements included in CPU modules 30 and 30' are the same.

Detailed Description Text (22):
CPU module 30 contains dual CPUs 40 and 50. CPUs 40 and 50 can be standard central processing units known to persons of ordinary skill. In the preferred embodiment described in the specification, CPUs 40 and 50 are VAX.RTM. processors manufactured by Digital Equipment Corporation, the Assignee of this application.

Detailed Description Text (24):
Serving as an interface between CPU 40 and cache 42 is a system support and cache control element 44, and serving as an interface between CPU 50 and cache 52 is a system support and cache control element 54. Elements 44 and 54 are identical and each provides a standard interface between the corresponding cache and also provides CPU and conventional peripheral functions such as interval timers. Cache busses 43 and 53 couple CPUs 40 and 50, respectively, to system support and cache control

modules 52 and 42, respectively.

Detailed Description Text (25):
2. Memory Module

Detailed Description Text (26):
Preferably CPUs 40 and 50 can share up to four memory modules 60. FIG. 5 is a block diagram of one memory module 60 and FIG. 6 is a detailed diagram showing specific memory elements of module 60.

Detailed Description Text (27):
Memory module 60 receives data from primary memory controller 70 via a 32 bit bidirectional memory bus 85. Memory module 60 also receives address/control signals from memory controllers 70 and 75 via busses 80 and 82, respectively. Busses 80 and 82 include row and column address signals as well as timing and control signals such as RAS (Row Address Strobe), CAS (Column Address Strobe), WE (Write Enable), and Refresh signals.

Detailed Description Text (28):
As shown in FIG. 5, memory module 60 includes a memory array 600. Memory array 600 is preferalby a standard RAM which is addressable by row and column addresses. In the preferred embodiment, memory array 600 can include up to eight banks of memory.

Detailed Description Text (31):
System timing/control signal generator 618 receives four types of inputs: clock signals; memory cycle signals, such as the write, read and refresh timing; certain other system control signals well known to persons of ordinary skill in the art; and address bit 29. Address bit 29 determines whether the address signals identify an access to memory space (i.e., in memory array 600), or I/O space (one of the I/O devices or system registers). System timing control signal generator 618 then controls the coordination and timing of the other elements of memory module 60 described below.

Detailed Description Text (32):
The configuration error log 620 shown in FIG. 5 stores information relevant to errors detected in the operation of memory module 60. Specifically, if an error is detected by compare logic 630, then configuration error logic 620 stores the necessary information to identify the faulty address and or data. Compare logic 630, however, only checks control and address signals, not memory data signals.

Detailed Description Text (34):
As shown in FIG. 6, configuration error logic 620, includes error processing logic 625 and EEPROM 626. Error processing logic 625 comprises an error counter, control logic and four storage registers, one for the primary address, one for the secondary address, one for the ECC, and one for a data word. Logic 625 generates error signals from the outputs of compare logic 630 as explained in detail below. Preferably, when an error condition is detected, the counter increments and the registers in logic 625 stores the primary and mirror memory addresses, ECC, and associated data word. EEPROM 626, which can be any type of NVRAM (nonvolatile RAM), stores memory error data for off-line diagnostics. When the memory module has been removed aftfer it faulted, the stored data is extracted from EEPROM 626 to determine the cause of the fault.

Detailed Description Text (37):
The four copies of each primary row and column address signals are inputs to address comparator 634, as are the mirror row and column address signals. In the preferred implementation of memory module 60, both the row and the column addresses are eleven bits long and are transmitted along busses 80 and 82 in alternate cycles. Thus, for each memory address there can be two sequential comparisons.

Detailed Description Text (44):
As explained above, error processing logic 625 also saves the primary and mirror address of the first ECC error that occurs during a read operation. An ECC error count in the counter in logic 625 is set to 1 for the first occurrence. Any subsequent ECC read errors will increment the ECC error count in memory module 60.

CPUs 40 and 50 periodically poll the stored address and count information in the memory module as part of their normal diagnostic testing. Part of the polling process clears these registers to allow the next address with an ECC error to be trapped. When CPUs 40 and 50 write corrected data to the trapped address these "soft errors" from memory array 600 are corrected.

Detailed Description Text (46):
As indicated above, memory module 60 does not perform any comparison of the data signals into memory. Primary and mirror memory controller 70 and 75 perform such a comparison. Memory controllers 70 and 75 control the access of CPUs 40 and 50, respectively, to memory module 60. The primary memory controller 70 is shown in greater detail in FIG. 7 and the mirror memory controller 75 is shown in greater detail in FIG. 8. Although memory controllers 70 and 75 are shown with slight differences in FIGS. 7 and 8, preferably they are identical for flexibility. These figures are drawn differently, simplify their explanation.

Detailed Description Text (47):
As shown in FIG. 7, primary control and address lines pass through primary memory controller 70 directly to memory module 60. Primary control signals on memory interconnect 80 are also processed and decoded by circuitry (not shown) to form all necessary timing and internal control signals, such as READ and WRITE.

Detailed Description Text (48):
Data lines 710 pass through write buffers 715 and 720 into memory interconnect 85 during write operation. During read operations, the data from memory module 60 on memory interconnect 85 passes through read buffer 725 and is an input to ECC generator 730 and ECC check/correct circuit 735. The output of ECC check/correct circuit 735 is an input to read buffer 740 whose output connects to data lines 710.

Detailed Description Text (49):
ECC generator 730 generates an ECC for the data received from data lines 710 to be written into memory module 60. The ECC from generator 730 is the primary ECC signal sent to memory module 60 through write buffer 745.

Detailed Description Text (50):
The primary ECC signal received from memory module 60 during a read operation is sent through read buffer 748 into ECC check/correct circuit 735. ECC check/correct circuit 735 checks the ECC generated from the data received from memory interconnect 85 to detect errors. Circuit 735 corrects single bit errors and sends the corrected data out through read buffer 740. If ECC check/correct circuit 735 determines that it cannot correct the error, then it sends an uncorrectable read error signal 738 to error latch 750 which stores that signal.

Detailed Description Text (52):
Another input into error latch 750 is the primary address/control error signal 762 from error processing and control logic 625. The remaining input into error latch 750 is a mirror miscompare signal 768. Mirror miscompare signal 768 is received from mirror memory controller 75 when its comparators detect a mismatch between the signals sent to memory module 60 from primary memory controller 70 and from mirror memory controller 75.

Detailed Description Text (54):
The details of mirror memory controller 75 are shown in FIG. 8. Mirror address and control signals 82 pass through and are decoded in mirror memory controller 75 just as the corresponding primary control signals 80 pass through primary memory controller 70. Data lines 711 are received through write buffer 716 and are an input to comparator 765. These data lines are also an input to ECC generator 731 which creates the mirror ECC signal. The mirror ECC signal is sent to memory module 60 by way of write buffer 746.

Detailed Description Text (55):
Data lines 711 may also be sent to memory module interconnect 85 via write buffer 722 if controller 75 needs to act as a primary controller. In general, however, only one set of data signals is sent to memory module 60, and buffer 722 is normally disabled.

Detailed Description Text (56):
Data is received into memory controller 75 on memory interconnect 85 from memory module 60 during both read and write operations. During write operations, the data on memory interconnect 85 is the same data as primary memory controller 70 sends to memory module 60. That data is received through write buffer 721 into comparator 765. If during write operations, the data from primary memory controller 70 is not equal to the data from mirror memory controller 75, then comparator 765 enables mirror miscompare signal 768 which is an input both to error latch 750 in primary memory controller 70 as well as to error latch 751 in mirror memory controller 75.

Detailed Description Text (57):
During read operations, data from memory module 60 is received from interconnect 85 through read buffer 726 and is then an input to ECC check/correct circuit 736. The mirror ECC signal received from memory module 60 is received through read buffer 749 and is also an input to ECC check/correct circuit 736. Similar to ECC check/correct circuit 735 in primary memory controller 70, ECC check/correct circuit 736 corrects all single bit errors prior to outputing data onto data line 711 through read buffer 741. If ECC check/correct circuit 736 cannot correct an error, it enables an uncorrectable read error signal 739 which is stored in error latch 751 in a manner similar to that of latch 750 in primary memory controller 70.

Detailed Description Text (58):
Error latch 751 also stores an ECC error signal 759 from AND gate 756, which combines mirror ECC error signal 753 from memory module 60 and the WRITE signal, and mirror address/control error signal 763 from error processing logic and control 625 in memory error module 60. The outputs of error latch 751 are inputs to OR gate 761. OR gate 761 enables a mirror memory error signal if any of the bits in error latch 751 are enabled.

Detailed Description Text (59):
Processing system 20' is a dual rail system internally. One rail includes CPU 40, cache memory 42, memory controller 70, and internal bus 46. The other rail includes CPU 50, cache memory 52, memory controller 75, and internal bus 56. Memory module 60, however, is a shared resource. The memory module 70 and 75 thus provide a dual rail-to-single rail interface for memory module 60. Thus, in accordance with the philosophy of this invention set forth in Section B of this description section, error checking is provided at this interface. In the preferred embodiment, such error checking involves two different techniques. First, the data signals from CPU 50 into memory controller 75 are not written into memory module 60, but are instead compared to the data signals from CPU 40 through memory controller 70. Memory controller 75 performs this comparison and an error check on the data sent to memory module 60. Memory module 60 compares the addresses, control signals and ECC's from memory controllers 70 and 75 to detect any inequality. The second error checking technique involves memory controllers 70 and 75 generating their own ECC from the memory data.

Detailed Description Text (60):
Another feature of this invention is the correction of single bit memory errors by memory controllers 70 and 75 instead by of having such errors cause a system fault. This technique accommodates single bit memory errors which are common, for example, from alpha particle bombardment. Correcting such errors reduces system fault time and allows the use of single, shared memory modules. Noting the occurrence and location of errors permits later diagnostics. For example, it may be desirable to replace a memory board which experiences more than a predetermined number of such correctable errors.

Detailed Description Text (61):
The interface between memory module 60 and both the primary and memory controllers 70 and 75 is shown generally at the left hand portion of FIGS. 7 and 8. FIG. 9 shows an interface circuit 770 of memory controller 70 with internal bus 46 and cross-link 90. An identical interface circuit is contained in memory controller 75.

Detailed Description Text (62):
Interface circuit 770 is also connected to a DMA engine 775 which provides address

and command signals for a direct memory access path to memory module 60. A detailed understanding of the general operation of DMA engine 775, which is preferably of conventional design, is not necessary for an understanding of the present invention. DMA engine 775 includes one counter with addresses for DMA transfer and another counter to keep track of the number of transfers. The address counter is incremented and the transfer number counter is decremented after each transfer.

Detailed Description Text (65):
In interface circuit 770, driver 780 provides a data path to internal bus 46 and CPU 40 when activated. Driver 780 is activated either for CPU reads of memory module 60 or reads of I/O. Multiplexer 792, which provides an input to driver 780, selects as the data for internal bus 40 either an input from buffer 788 if CPU 40 is reading memory, or an input from buffer 790 if CPU 40 is reading data from an I/O device.

Detailed Description Text (66):
Driver 782 provides a data path to memory controller 70 and is activated either for CPU writes to memory module 60, DMA writes to memory module, or memory resync (slave) operations. Memory resync operations are described in detail below. Those operations are used to ensure that the contents of memory modules 60 and 60' are set equal to each other. In memory resync operations, the module receiving data is the "slave" and the module sending data is the "master". Multiplexer 794, which provides an input to driver 782, selects as the data for memory module 60 either an input from buffer 786 if the operation being performed is a CPU memory write, or an input from buffer 790 if the operation is either a DMA write or a memory resync (slave) operation.

Detailed Description Text (70):
Data for memory resync, DMA and I/O operations pass through cross-links 90 and 95. Generally, cross-links 90 and 95 provide communications between CPU module 30, CPU module 30', I/O modules 100, 110, 120, and I/O modules 100', 110', 120'. Since cross-links 90 and 95 are identical, only the elements and operation of cross-link 90 will be described.

Detailed Description Text (74):
The cross-link can produce an interrupt to both zones and can be for reception of data by the other zone or for making data available for the present zone. The DCR can only be used when the I/O modules are in lock-step and the cross-links are in a master/slave mode. One zone, the master zone, controls the I/O device of the other zone, the slave zone, and the cross-link in the slave zone is used for communications. The cross-link DCR in the master zone is not used. The interrupts generated go to both the slave zone CPUs and to the master zone CPUs. Each zone makes its own choice whether to use interrupts or polling for transmitting and receiving data.

Detailed Description Text (88):
"Cross-link off" refers to the state when no communication between 20 and 20' is allowed with the parallel cross-link 910. This mode assumes no synchronization between the zones. "Cross-link on slave" refers to the state a zone uses to give its module interconnect to the other zone. If zone A is in "cross-link on slave," zone B wll be in "cross-link on master." Thus CPUs 40' and 50' will have control over module interconnects 130 and 132 as well as 130' and 132'. CPUs 40 and 50 have no access to interconnects 130, 132, 130', and 132." "Cross-link on slave" and "cross-link on master" assume synchronization between module interconnects but not between CPUs.

Detailed Description Text (89):
"Cross-link on duplex" allows both CPUs 40 and 40' to control module interconnects 130 and 130' and CPUs 50 and 50' to control interconnects 132 and 132'. "Cross-link on duplex" assumes synchronization of CPUs as well as synchronization of I/O interconnects. Synchronization of CPUs requires that all memory registers and bus cycles of both zones be identical. Any uniqueness of data must be handled through the communication register 906. Synchronization of the module interconnect implies that cross-links 90, 95, 90', and 95' are driving their respective module interconnects 130, 132, 130' and 132' with same or compatible bus cycle.

Detailed Description Text (90):
When cross-link 90 is in "master mode", CPUs 40 and 50 are controlling all four module interconnects 130, 132, 130' and 132'. In order for CPUs 40 and 50 to check the status of cross-link 90' and 95', status register 909's read and write addresses are in the system address space.

Detailed Description Text (111):
Controller 925 of cross-link 90 is shown in FIG. 12. Control decoder 930 in controller 925 produces signals A-I according to rules set forth in detail below. Control 925 includes drivers, buffers, multiplexers, and delay elements. The delay elements are added for synchronization. Because of the high speed operation of computer system 10 as well as the tight synchronization requirements, the cross-links must compensate for inherent signal delays such as cable runs to maintain proper operation and synchronization. For example, as explained below in greater detail, during a read from I/O, each CPU module receives data from I/O modules in different processing systems 20 and 20'. Because data from the different systems take different paths, such operations could cause a failure to synchronize. The delay elements compensate for the signal delays and path differences to preserve synchronization. As shown in FIG. 12, the delay elements in cross-link 90 are used to slow down an input to a multiplexer when the other input is from parallel cross-link pathway 25.

Detailed Description Text (112):
In controller 925, driver 933 is enabled by signal A to transmit data to memory controller 70. Driver 936 is enabled by signal B to transmit data to the parallel registers by way of parallel register bus 910. Driver 939 is enabled by signal C to transmit data to module interconnect 130. Driver 942 is enabled by signal D to transmit data to parallel cross-link pathway 25.

Detailed Description Text (115):
Multiplexer 963 is controlled by signal F and receives inputs from parallel register bus 910 via buffer 957, from the serial registers, and from module interconnect 130 via buffer 954. The output of multiplexer 963 is an input to multiplexer 960 after passing through delay element 980.

Detailed Description Text (118):
Multiplexer 970 is controlled by signal I and receives inputs from the parallel registers via bus 910 and buffer 957, from memory controller 70 via buffer 945, and from module interconnect 130 via buffer 954. The output of multiplexer 970 is sent to the parallel cross-link pathway 25 by way of driver 942.

Detailed Description Text (132):
FIG. 13A shows the state of the control signals for cross-link 90 (i.e., on the primary rail in zone 11) when in a duplex mode. For a read I/O operation, control signals A and D are "on" and B anc C are "off." Signal A enables driver 933 which ensures that data will pass to memory controller 70, and signal D enables driver 942 ensuring the data will also pass to cross-link 90 via parallel cross-link pathway 25. The multiplexers which are involved in sending data to memory controller 70 and to parallel cross-link pathway 25 are multiplexers 960, 963 and 970 which are controlled by signals E, F and I, respectively. Control signal E is set to select input 1 which corresponds to the output of multiplexer 963. Control signal F is set so that multiplexer 963 selects input 3, which corresponds to data from module interconnect 130. Control signal I is set so that multiplexer 970 also selects the module interconnect 130.

Detailed Description Text (133):
With this signal selection, data from module interconnect 130 thus passes through multiplexers 963 and 960, as well as driver 933, to memory controller 70, which is the data path for a Read I/O transaction. That data also passes through multiplexer 970 and driver 942 onto parallel cross-link pathway 25, which is appropriate since cross-link 90 is in the duplex mode.

Detailed Description Text (134):
FIG. 13E shows the control signals for the identical situations, except that memory controller 90 is in the master mode and thus is not sending signals to cross-link

90'. The Read I/O operation in this situation involves the same signal settings to ensure that data from module interconnect 130 passes to memory controller 70. Since signal D is "off" and there is no selection signal for I, there is no data pathway to memory controller 90' via parallel cross-link pathway 25, which is appropriate for operation when cross-link 90 is in the master mode.

Detailed Description Text (135):
For a Write I/O operation with memory controller 90 in duplex mode, the only driver control signal that is "on" is signal C which enables driver 939 to pass data to module interconnect 130. Therefore, the only multiplexer that needs to be controlled is multiplexer 969, and FIG. 13A indicates that control signal H selects data from memory controller 70. Note that the operation of cross-link 95 (mirror rail in zone 11 in duplex mode) FIG. 13 differs slightly from the control signals for cross-link 90. (FIG. 13B). This is because, as explained in detail below, during a write I/O operation, data to each I/O module comes from both processors 20 and 20'.

Detailed Description Text (137):
When both processing systems 20 and 20' are performing the same tasks in a redundant fashion, known as the duplex mode, it is imperative that CPU modules 30 and 30' perform operations at the same rate. Otherwise, massive amounts of processing time will be consumed in resynchronizing the processing systems for I/O and interprocessor error checking. In the preferred embodiment of processing systems 20 and 20', their basic clock signals are synchronized and phase-locked to each other. The fault tolerant computing system 10 includes a timing system to control the frequency of the clock signals to processing systems 20 and 20' and to minimize the phase difference between the clock signals for each processing system.

Detailed Description Text (138):
FIG. 14 shows a block diagram of the timing system of this invention embedded in processing systems 20 and 20'. The timing system comprises oscillator system 200 in CPU module 30 of processing system 20, and oscillator system 200' in CPU module 30' of processing system 20'. The elements of oscillator 200' are the same as those for oscillator 200 and both oscillator systems operation is the same. Thus, only the elements and operation of oscillator 20 will be described, except if the operations of oscillator 200 and 200' differ.

Detailed Description Text (147):
6. I/O Module

Detailed Description Text (148):
FIG. 17 shows a preferred embodiment of an I/O module 100. I/O modules 100 and 100' are identical so only module 100 is described. In addition, I/O modules 110 and 120 (and thus modules 110' and 120') are similar to module 100, but it is unlikely that they would have exactly the same configuration due to their connection to different I/O devices.

Detailed Description Text (149):
I/O module 100 is connected to CPU module 30 by means of dual rail module interconnects 130 and 132. Each of the module interconnects is received by firewalls 1000 and 1010, respectively. Firewalls 1000 and 1010 are interconnected by a checking bus 1005 which checks the equality of the data transmitted on module interconnects 130 and 132. That checking is effective due to the lock step synchronization of CPU modules 30 and 30' which cause data written to I/O module 100 from CPU modules 30 and 30' to be available at firewalls 1000 and 1010 simultaneously.

Detailed Description Text (150):
FIG. 18 shows the elements and the preferred embodiment of firewall 1000. Firewall 1000 includes a 32 bit bus interface 1810 to module interconnect 130 and a 32 bit bus interface 1820 for connection to bus 1020 shown in FIG. 17. Interfaces 1810 and 1820 are interconnected by an internal firewall bus 1815 which also interconnects the other elements of firewall 1000. Preferably bus 1815 is a parallel bus either 16 or 32 bits wide.

Detailed Description Text (151):

Firewall 1000 also preferably includes console support registers 1830 which connect to a console link 1090 between CPU module 30 and operator consoles. The console link bypasses the normal data paths, such as module interconnects, cross-links and memory controllers so that operators can communicate with the CPUs even if there are faults in other parts of computer system 10. Console support registers 1830 preferably include temporary storage registers 1832, a receiver 1834, and a transmitter 1836. The interconnection of registers 1832, receiver 1834, and transmitter 1836 are conventional and known to persons of ordinary skill.

Detailed Description Text (152):
Firewall 1000 also includes a firewall comparison circuit 1840 which includes a plurality of checkers. Firewall comparison circuit 1840 is connected to an equivalent element in firewall 1010. The checkers in firewall comparison circuit 1840 compare the data received from module interconnect 130 with the data received from module interconnect 132. In general, the data from module interconnect 130 is from one zone and the data from module interconnect 132 is from another zone, even though both module interconnects are received from CPU module 30. If the checkers in firewall comparison circuit 1840 detect any inequality between those data, interrupt control circuit 1880 is activated to generate an error signal which causes cross-links 90 and 95 to notify CPU modules 30 and 30' of a fault detection.

Detailed Description Text (153):
Firewall comparison circuit 1840 only checks data received from CPU modules 30 and 30'. Data sent to CPU modules 30 and 30' has a common origin and thus does not require checking. Instead, data received from an I/O device to be sent to CPU modules 30 and 30' is checked by an error detection code (EDC), such as a cyclical redundancy check (CRC) which is performed by CRC generator 1850. CRC generator 50 is also coupled to internal firewall bus 1815.

Detailed Description Text (154):
CRC generator 1850 generates and checks the same CRC code that is used by the I/O device. Preferably, I/O module 100 generates two EDC's. One, which can also be a CRC, is used for an interface to a network, such as the Ethernet packet network to which module 100 is coupled. The other is used for a disk interface such as disk interface 1072 in FIG. 17.

Detailed Description Text (155):
CRC coverage is not required between CPU module 30 and I/O module 100 because the module interconnect is duplicated. For example in CPU module 30, cross-link 90 communicates with firewall 100 through module interconnect 130, and cross-link 95 communicates with firewall 1010 through module interconnect 132.

Detailed Description Text (156):
A message received from Ethernet network 1082 is checked for a valid CRC by network control 1080 shown in FIG. 17. The data, complete with CRC, is written to a local RAM 1060 also shown in FIG. 17. All data in local RAM 1060 is transferred to memory module 60 using DMA. A DMA control 1890 coordinates the transfer and directs CRC generator 1850 to check the validity of the CRC encoded data being transferred.

Detailed Description Text (158):
DMA control 1890 controls the operation of CRC generator 1850 relative to the data transfer. When the logical block has been transferred, DMA control 1890 reads the generated CRC from the CRC generator 1850 and appends it to the data stored in local RAM 1060. When network control 1080 transfers the data from local RAM 1060 to the Ethernet network 1082, it checks the CRC. All of the Ethernet packet except the CRC code itself is transferred to memory module 60. Any errors in the CRC will be indicated by the CRC generator 1850 and will be reported through Interrupt Control 1880.

Detailed Description Text (159):
The data transfers to and from the disk subsystem occur in a manner analogous to the Ethernet interface. The CRC generator 1850 generates or checks the specific CRC code used by the disk control 1072. This ensures that data residing in or being transferred through a single rail system like I/O Module 100 is covered by an error detection code, which is preferably at least as reliable as the communications media

the data will eventually pass through. Different I/O modules, for example those which handle synchronous protocols, preferably have a CRC generator which generates and checks the CRC codes of the appropriate protocols.

Detailed Description Text (161):
The I/O modules (100, 110, 120) are responsible for controlling the read/write operations to their own local RAM 1060. The CPU module 30 is responsible for controlling the tranfer operations with memory array 60. The DMA engine 775 of memory controllers 70 and 75 (shown in FIG. 9) directs the DMA operations on the CPU module 30. This division of labor prevents a fault in the DMA logic on any module from degrading the data integrity on any other module in zones 11 or 11'.

Detailed Description Text (162):
Firewall 1000 also performs other key functions for I/O module 100. An I/O Diagnostic Control Register 1860 in firewall 1000 has indentical construction to the cross-link Diagnostic Control Register 901 and also allows communication between CPUs 40, 50, 40', and 50' and diagnostic microprocessor 1100. This indirect connection of diagnostic microprocessor 1100 prevents that microprocessor from affecting any other module in computer system 10.

Detailed Description Text (163):
The functions of trace RAM 1872 and trace RAM controller 1870 are described in greater detail below. Briefly, when a fault is detected and the CPUs and CPU modules 30 and 30' are notified, then various trace RAMs throughout computer system 10 are caused to perform certain functions described below. The communications with the trace RAMs takes place over trace bus 1095. Trace RAM control 1870, in response to signals from trace bus 1095, causes trace RAM 1872 either to stop storing, or to dump its contents over trace bus 1095.

Detailed Description Text (164):
I/O Module Bus 1020, which is preferably a 32 bit parallel bus, couples to firewalls 1000 and 1010 as well as to other elements of the I/O module 100. A shared memory controller 1050 is also coupled to I/O bus 1020 in I/O module 100. Shared memory controller 1050 is coupled to a local memory 1060 by a shared memory bus 1065, which has 32 bits plus parity. Preferably, local memory 1060 is RAM with 128 KB of memory, but the size of RAM 1060 is discretionary. The shared memory control 1050 and local RAM 1060 provide memory capability for I/O module 100.

Detailed Description Text (165):
Disk controller 1070 provides a standard interface to a disk, such as disks 1075, 1075' in FIG. 1. Disk 1075 is preferably connected to disk controller 1070 by a standard bus interface 1072, such as an implementation of the SCSI (small computer standard interface) bus. Disk controller 1070 is also coupled to shared memory controller 1050 either for use of local RAM 1060 or for communication with I/O module bus 1020.

Detailed Description Text (166):
A network controller 1080 provides an interface to a standard network, such as the ETHERNET network, by way of network interface 1082. Network control 1080 is also coupled to shared memory controller 1050 which acts as an interface both to local RAM 1060 and I/O module bus 1020. There is no requirement, however, for any one specific organization or structure of I/O Module Bus 1020.

Detailed Description Text (167):
PCIM (power and cooling interface module) support element 1030 is connected to I/O module bus 1020 and to an ASCII interface 1032. PCIM support element 1030 allows processing system 20 to monitor the status of the power system (i.e., batteries, regulators, etc.) and the cooling system (i.e., fans) to ensure their proper operation. Preferably, PCIM support element 1030 only receives messages when there is some fault or potential fault indication, such as an unacceptably low battery voltage. It is also possible to use PCIM support element 1030 to monitor all the power and cooling subsystems periodically. PCIM support element 1030, as well as an equivalent element in I/O module 100', enables fault tolerant computing system 10 to shut down a zone in case of malfunctions of the support systems for processors 20 and 20'.

Detailed Description Text (168):
System support and console element 1040 is also coupled to I/O module bus 1020. System support and console element 1040 provides an interface for an operator's console via ASCII interface 1042. The operator's console not only allows input of certain information, such as time of year, the console may also be used for diagnostic purposes. The operator console exchanges data with the CPUs over console link 1090 and 1091 via console support registers 1830 in firewalls 1000 and 1010.

Detailed Description Text (169):
Diagnostics microprocessor 1100 is also connected to the I/O module bus 1020. The operation of the diagnostics microprocessor 1100 are described in detail below. In general, microprocessor 1100 is used to gather error checking information from trace RAMS, such as trace RAM 1872, when faults are detected. That data is gathered into trace buses 1095 and 1096, through firewalls 1000 and 1010, respectively through, module bus 1020, and into microprocessor 1100.

Detailed Description Text (172):
The elements of computer system 10 do not by themselves constitute a fault tolerant system. There needs to be a communications pathway and protocol which allows communication during normal operations and operation during fault detection and correction. Key to such communication is cross-link pathway 25. Cross-link pathway 25 comprises the parallel links, serial links, and clock signals already described. These are shown in FIG. 19. The parallel link includes two identical sets of data and address lines (16), control lines (7), interrupt lines (7), and error lines (7). The sixteen data and address lines and the seven control lines contain information to be exchanged between the CPU modules, such as from the module interconnects 130 and 132 (or 130' and 132') of from memory module 60 (60').

Detailed Description Text (174):
The fault tolerant processing system 10 is designed to continue operating as a dual rail system despite most transient faults. The I/O subsystem (modules 100, 110, 120, 100', 110', 120') can also experience transient faults and continue to operate. In the preferred embodiment, an error detected by firewall comparison circuit 1840 will cause a synchronized machine check to be requested through lines 25 for CPU directed operations. Software in CPU 30 and 30' will retry the faulted operation. For DMA directed operations, the same error indicated by error signal detection results in synchronous interrupts through line 25, and software in CPUs 40, 50, 40' and 50' will restart the DMA operation.

Detailed Description Text (175):
Certain transient errors are not immediately recoverable. For example, a control error 762 in CPU module 30 can result in unknown data in memory module 60. In this situation, CPU module 30 can no longer function reliably as part of a fail safe system so it is removed. Memory array 60 must then undergo a memory resync before CPU 30 can rejoin the system. The CPU/memory failure line indicates to CPU 30' that CPU 30 has been faulted.

Detailed Description Text (176):
The seven control lines, which represent a combination of cycle status, byte mask, direction, and ready conditions, provide the handshaking between CPU modules (30 and 30') and the I/O modules. Cycle status describes the type of bus operation being performed: CPU read of I/O, DMA transfer, DMA setup, or interrupt vector request. "Byte mask" directs which of the 16 data lines contains valid data to allow modification of a byte (8 bits) of data in a 32 bit word in local RAM 1060. "Direction" fixes the selection of transceiver paths for DMA operation. Since DMA transfers occur between a predetermined source and destination, the time overhead associated with selecting the data paths need not be paid for every data transfer. "Ready" messages are sent between the CPU and I/O modules to indicate the completion of requested operations.

Detailed Description Text (180):
FIGS. 20A-D show block diagrams of the elements of CPU modules 30 and 30' and I/O modules 100 and 100' through which data passes during the different operations. Each of those elements has each been described previously.

Detailed Description Text (181):
In general, the data paths during the operations are symmetric. For example, a CPU I/O read proceeds to I/O module 100 as well as I/O module 100' over paths which are mirror images of each other. When such is the case, the figures show all data paths, but only one is described in detail.

Detailed Description Text (182):
FIG. 20A shows the data pathways for a CPU I/O read operation. Data, for example either from a disk 1075 (1075') or a network, are presumed to be stored in local RAM 1060 (1060') for transfer through shared memory controller 1050 (1050'). For one path, the data pass through firewall 1000, module interconnect 130, to cross link 90. At that time, the data are transferred up to memory controller 70 and to cross-link 90'. As seen in FIG. 12, cross-link 90 delays the data from firewall 100 to memory controller 70 so that the data to cross-link 90' have enough time to "catch up" and processing systems 20 and 20' remain synchronized. The data then proceeds out of cross-link 90 through memory controller 70 and into CPU 40 by way of internal bus 46. Similarly, the data to cross-link 90' proceeds to CPU 40'.

Detailed Description Text (185):
Although I/O modules 100, 110, and 120 are similar and correspond to I/O modules 100', 110', and 120', respectively, the corresponding I/O modules are not in lock step synchronization. Using memory controller 1050' and local RAM 1060' for CPU I/O read, the data would first go to cross-links 90' and 95'. The remaining data path is equivalent to the path from memory controller 1050. The data travel from the cross-links 90' and 95' up through memory controllers 70' and 75' and finally to CPUs 40' and 50', respectively. Simultaneously, the data travel across to cross links 90 and 95, respectively, and then, without passing through a delay element, the data continues up to CPUs 40 and 50, respectively.

Detailed Description Text (190):
As with the CPU I/O write operation, this data path as well as the symmetric paths through firewalls 1000' and 1010' provide interzonal error checking. Interrail error checking occurs in memory modules 70, 75, 70' and 75'.

Detailed Description Text (194):
As FIG. 21 shows, the first step in the bulk memory copy operation involves setting the cross-links to the memory resync master/slave mode to permit the data path shown in FIG. 20E (step 2100). FIGS. 13I-13L show that when a cross-link is in the slave mode, the communication is essentially one way. Thus, data can be written into the memory of the processing system whose cross-links are in the slave mode, but information may not be read out of that memory. Furthermore, FIGS. 13E-13H show, in conjunction with 13I-13L, that every time the processing system whose cross-link is in the master mode writes into its memory module, the processing system whose cross-link is in the slave mode, also writes that same data. Thus, the processing system whose cross-link is in the slave mode has a memory module which is storing up-to-date information along with the processing system whose cross-link is in the master mode.

Detailed Description Text (195):
Next, as shown in the bulk memory transfer flow chart of FIG. 21, the master processing system audits its memory module to find out all of the memory pages in that module (step 2110). Once all those pages are known, they are queued into the DMA engine of the master processing system (step 2120). Preferably, the DMA engine of the master processing system includes a queue of DMA requests, each of the requests preferably includes an indication of the type of transfer, the starting address, the number of elements to be transferred, and an indication of the destination. The destination for all DMA transfers to the slave processing system is the memory module in the slave processing system.

Detailed Description Text (196):
The memory resync operation then begins an iterative process of sending commands to the DMA engine in the slave processing system (step 2130) and initiating and completing DMA transfers of the next page (step 2140). The command to the slave DMA engine indicates the size of the DMA transfers and the starting addresses to ensure

that the slave processing system's memory module is configured to be identical to that of the master module. These steps are repeated until the transfer of all the master memory is complete (step 2150).

Detailed Description Text (201):
FIGS. 22A-22H contain a flowchart showing a bootstrapping sequence to bring CPU modules 30 and 30' into lock step synchronization.

Detailed Description Text (213):
Zone A then waits until the DMA is complete for memory resync (step 2268), reads its system state from memory (step 2270), and saves the restart vector, which is the memory location from which the zones will start after resync (step 2272). The system state is all the register resident information in the CPU module that controls the execution of the software and hardware. This information includes, but is not restricted to, CPU general purpose registers, address translation tables, process status words, stack pointers, program counters, interrupt status, configuration registers, and interval timers. These values are saved in memory array 60. Because memory resync is still in effect, the values are also saved in memory array 60'. The memory address of the saved values is written into the save state registers in the cross-links. All processing in zone A is then suspended and cache memory is flushed. All information necessary to restart application processing is resident in memory array 60 and 60'.

Detailed Description Text (230):
D. Fault Detection Isolation and Repair

Detailed Description Text (232):
Different methods for fault detection have already been discussed in the explanation of CPU modules 30 and 30' and firewalls 1000 and 1010. Those methods include checks by memory modules 60 and 60' that the address and control signals received from memory controllers are identical and checks for uncorrectable memory data errors. The methods also include comparisons by memory controller pairs 70/75 and 70'/75' that address, control, and data signals passing through the two "rails" are the same. Also firewalls 1000 and 1010 have checkers which compare data from zones during I/O writes. Other fault detection schemes are also possible.

Detailed Description Text (233):
Consistent with the fail stop capability of processing system 20 and 20', the fault detection in the processing systems keeps faults from propagating to other modules in an uncontrolled manner. To realize a fail safe capability, however, computing system 10 must isolate the source of the fault so that the system can be repaired. The preferred techniques for locating the source of faults uses recirculating registers, called trace RAMs, located throughout data paths in processing systems 20 and 20'. These trace RAMs record consecutive messages transmitted on the associated data paths. When a fault occurs, the messages are analyzed to locate the source of the fault.

Detailed Description Text (234):
FIG. 25 is a block diagram of CPU module 30 and I/O module 100 showing preferred locations of trace RAMs in computing system 10. Other locations may also be used. In processing system 20, trace RAMs are preferably located on all data paths. Thus in FIG. 25, trace RAMs 2500 and 2505 are located on memory controllers 70 and 75, respectively. Trace RAMs 2510, 2515, and 2518 are located on all the interfaces of cross-link 90 and trace RAMs 2520, 2525, and 2528 are located on all the interfaces of cross-link 95. Trace RAMs 1872 and 1877 are located in firewalls 1000 and 1010, respectively. A complementary set of trace RAMs are located in processing system 20'.

Detailed Description Text (238):
Each of the trace RAMs then keeps in its memory a copy of the N most recent transactions on the data pathway associated with it. For example, in FIG. 25 trace RAM 2518 keeps a copy of the N most recent transactions on module interconnect 130.

Detailed Description Text (240):
FIG. 27 is a flow chart for the procedure used for isolating the source of errors or

faults once they are detected. Errors can either be "intermittent," which do not repeat "solid," which do repeat. Detection of a fault by some element causes an error signal to be sent to the associated cross-link (i.e., cross-link 90 for firewall 1000 and memory controller 70). The cross-links either interrupt the CPUs or cause them to enter into a trap.

Detailed Description Text (243):
The CPUs service the trap or interrupt by entering the machine check routine (step 2702) and determining whether the error was detected while the CPUs were already in the machine check routine (step 2705). If so, then the error is assumed to be a solid error and the system begins the procedures for isolating solid faults shown in FIGS. 28A-C.

Detailed Description Text (244):
If the CPUs were not in a machine check procedure, then the detected error is assumed to be intermittent. The CPUs then set an entry flag (step 2710) to begin procedures for isolating that error. The entry flag is the flag tested in step 2705 to determine whether the CPUs were in a machine check procedure when interrupted.

Detailed Description Text (246):
Each CPU then starts at the element which sent the error signal (step 2740) to isolate the source of the fault at an element. Comparison of Trace RAM data is done by comparing the data from Trace RAMs at the same locations on opposite rails, such as 2500/2505, 2510/2520, 2515/2525, etc. The corresponding messages (i.e., those at the same depth in the trace RAM) are compared to see whether an error (i.e, inequality between corresponding messages) is detected. After the trace RAMs at the error detector are analyzed, the next trace RAMs follow, in reverse, the path data followed in the operation during which the fault occurred. Thus, for CPU reads, the path goes away from the CPU's.

Detailed Description Text (247):
In the first step of the error detection routine, the trace RAM data from each rail of CPU module 30 and 30' are compared (step 2750). If there are no errors between the trace RAMs on different rails, (step 2752), meaning that the trace RAM data on corresponding data paths are equal, the presence of other data paths is determined (step 2754). If other data paths remain, an alternate path is taken (step 2756) for subsequent comparisons (steps 2750, et seq.).

Detailed Description Text (252):
Once a fault has been determined to be a solid fault (see FIG. 27), the procedure for isolating such solid faults, shown in the flow charts in FIG. 28A-28C, is initiated. The first step in the solid fault processing involves splitting the fault tolerant computing system 10 into its two separate zones 11 and 11' (step 2800). This is done by turning all cross-links off. The diagnostic microprocessors in each zone then read the trace RAM data for their zone (step 2805) and send that data to the zone's CPUs via a dual rail access (step 2810). If an error is detected during this process (step 2815) then a hardware fault has occurred during the error processing, and that zone is removed from computer system 10 (step 2820). An attempt is made to report the fault (step 2825), which may not be possible because of the hardware fault, and the procedure exits to the console. This allows the console operator to initiate detailed diagnostic tests to locate the solid fault.

Detailed Description Text (254):
If the zone's analysis of the trace RAM data shows that it has an error (step 2835), then the error location is examined (step 2840). If the error is internal to the zone, then the fault is determined to be no longer hard or solid, or multiple faults are found to be present a transient fault occurring while attempting to isolate a transient fault will make the original transient appear to be solid (step 2845). Although rare, this condition requires further analysis so, similar to the steps followed if there is a hardware fault during error processing, the faulting zone is removed from the system (step 2850), an attempt is made to report the error (step 2855), and the solid fault procedure is exited to the console.

Detailed Description Text (262):
The results of the fault and solid fault isolation procedures in FIGS. 27 and 28A-C,

could be that an error or fault has been located in a CPU module (i.e., the CPU/memory/memory controller combination); a cross-link; cross-link pathway 25; a module interconnect; or an I/O module. Generally, if an intermittent fault is detected, the CPUs in fault tolerant computer system 10 note the location of the intermittent fault, increment counters indicating the number of intermittent errors for the faulting element, and takes no further action unless the count for an element exceeds a predetermined number. If the count does exceed that predetermine number, indicating that the frequency with which intermittent faults occur is too high to be tolerated, or if the detected fault is a solid fault, then the module or element is defective, and must be disabled so that the fault tolerated computer system 10 can continue to operate effectively until the module containing that element can be replaced.

Detailed Description Text (263):
If a CPU module is found to be at fault, then that module must be disabled by switching the cross-link of the module from the "duplex" mode to the "slave" mode. In a "slave" mode, a module can only communicate with the CPU module in the other zone by way of its cross-link diagnostic control register. The disabled CPU module is then returned to the console module.

Detailed Description Text (264):
If a cross-link is found at fault, then one of three actions may be taken depending upon the portion of the cross-link at fault. The portion of the cross-link coupled to the memory controller is virtually indistinguishable from the associated CPU module. If that portion is faulty, then the same procedures for disabling the CPU module must occur.

Detailed Description Text (265):
The portion of the cross-link coupled to the module interconnect is virtually indistinguishable from the module interconnect. The process of disabling the cross-link in this situation is the same as that for disabling the module interconnects.

Detailed Description Text (266):
The portion of the cross-link is coupled to cross-link pathway 25 virtually indistinguishable from the cross-link pathway. In this case, or if the fault is in the cross-link pathway, the zones are split apart by turning are cross-links off and disabling the CPU module believed to contain the faulty cross-link. If the source of the fault cannot be determined, however, then a preselected one of the CPU modules 30 and 30' is disabled first. If errors persist the other CPU module is rebooted and the preselected module is disabled.

Detailed Description Text (267):
The same procedure is used to disable faulty I/O modules, module interconnects, or cross-links where the faults occur in the module interface. In all three instances, the fault occurred during a transaction involving an I/O module, so that I/O module is disabled. This is done by advising the operating systems of all of the CPU's 40, 50, 40', and 50' to remove reference to that I/O module.

Detailed Description Text (268):
If the fault recurs, indicating a cross-link or module interconnect fault, the zone with those modules can be disabled. The preferable solution is to keep disabling the I/O module in communication when the fault occurred. The module interconnects and the associated portion of the cross-links are effectively disabled when all the connected I/O modules are disabled. A CPU module with no I/O modules is effectively disabled for most purposes.

Other Reference Publication (21):
Su et al., "A Hardware Redundancy Reconfiguration Scheme for Tolerating Multiple Module Failures," IEEE Transactions on Computers, vol. C-29, No. 3 (Mar. 1980).

CLAIMS:

1. A fault tolerant computer system comprising:

a fault tolerant data processing module including

means for detecting and correcting errors in the operation of said data processing module to maintain a high degree of data integrity,

data transmission control means for controlling the transmission of all data to said fault tolerant data processing module and for controlling the receipt of all data into said fault tolerant data processing module, and

input/output terminals, coupled to said data transmission control means, for receiving and transmitting said data; and

a non-fault tolerant input/output module, coupled to transmit said data to said input/output terminals of said fault tolerant data processing module, said non-fault tolerant input/output module including

read means for transferring data to said fault tolerant computing system in response to requests from said data transmission control means, and

firewall means for preventing said non-fault tolerant input/output module from initiating transfers of data to said fault tolerant data processing module.

2. The fault tolerant computer system of claim 1 wherein

said fault tolerant data processing module includes

a memory for storing data from said input/output module; and

address means, coupled to said memory and under sole control of said fault tolerant data processing module, for generating addresses indicating the locations in said memory where said data is to be stored.

3. The fault tolerant computer system of claim 1 wherein said input/output module includes an input/output bus for transferring information internal to said input/output module, and wherein said firewall means includes a bus interface element coupled to said input/output bus.

4. The fault tolerant computer system of claim 1 wherein said data transmission control means includes a cross-link element to control communication between said fault tolerant computer system and said non-fault tolerant input output module.

5. The fault tolerant computer system of claim 4, wherein said fault tolerant data processing module includes first and second central processing units, and

wherein said cross-link element includes

first means for routing said data between said first and second central processing units, and

second means for routing said data between said first and second processors and said non-fault tolerant input/output module.

6. A fault tolerant computer system comprising:

a fault tolerant data processing module including

means for detecting and correcting errors in the operation of said data processing module to maintain a high degree of data integrity,

a memory unit for storing data for said fault tolerant data processing module,

data transmission control means for controlling the receipt of all input data into said fault tolerant data processing module, said data transmission control means including means for generating all addresses in said memory unit for locations in said memory unit into which data is to be stored, and

input terminals, coupled to said data transmission control means, for receiving said input data; and

an output module, coupled to transmit said input data to said input terminals of said fault tolerant data processing module, said output module including output means for transmitting said input data to said fault tolerant data processing module.

7. The fault tolerant computer system of claim 6

wherein said output module further includes storage means for storing data received from said fault tolerant data processing module, and

storage control means, coupled to said storage means, for generating all addresses in said storage means into which data is to be received from said fault tolerant data processing module; and

wherein said fault tolerant data processing module includes

an output terminal, coupled to said output module, for transmitting data to said output module, and

means, coupled to said output terminal for transmitting data to said output terminal for transmission to said output module.